Performing Deep Recurrent Double Q-Learning for Atari Games

Felipe Moreno-Vera

Performing Deep Recurrent Double Q-Learning for Atari Games

Felipe Moreno-Vera
felipe.moreno@ucsp.edu.pe

Universidad Catlica San Pablo, Arequipa, Perú

Abstract. Currently, many applications in Machine Learning are based on define new models to extract more information about data, In this case Deep Reinforcement Learning with the most common application in video games like Atari, Mario, and others causes an impact in how to computers can learning by himself with only information called rewards obtained from any action. There is a lot of algorithms modeled and implemented based on Deep Recurrent Q-Learning proposed by DeepMind used in AlphaZero and Go. In this document, We proposed Deep Recurrent Double Q-Learning that is an implementation of Deep Reinforcement Learning using Double Q-Learning algorithms and Recurrent Networks like LSTM and DRQN.

Keywords: Deep Reinforcement Learning · Double Q-Learning · Recurrent Networks · Convolutional Networks · Reinforcement Learning · Atari · Video Games · DQN · DRQN · DDQN

1 Introduction

Currently, there is an increase the number of application in Reinforcement Learning, specially in Deep Reinforcement Learning with new techniques. One of application of DRL (Deep Reinforcement Learning) is in Games like AlphaZero (Go, Chess, etc) and video games like Mario, Top racer, Atari, etc. Deep Reinforcement Learning is considered like a third model in Machine Learning (with Supervised Learning and Unsupervised Learning) with a different learning model and architecture.

There are several methods of implementing these learning processes, where Q-Learning is a prominent algorithm, the Q value of a pair (state, action) contains the sum of all these possible rewards. The problem is that this sum could be infinite in case there is no terminal state to reach and, in addition, we may not want to give the same weight to immediate rewards as to future rewards, in which case use is made of what is called an accumulated reinforcement with discount: future rewards are multiplied by a factor γ [0, 1] so that the higher this factor, the more influence future rewards have on the Q value of the pair analyzed. Formally:

Richard Sutton [1, 2] define various models to describe Reinforcement Learning and how to understand it. DeepMind was the first to achieve this Deep Learning with AlphaZero and Go game using Reinforcement Learning with Deep Q-Learning (DQN) [3] and Deep Recurrent Q-Leaning (DRQN) [4], follow up by OpenAI who recently suprased professional players in Star Craft 2 (Gramve created by Blizzard) and previously in Dota 2 developed by Valve. Chen et al. [5] proposed a CNN based on DRQN using Recurrent Netowrks (a little variance of DRQN model using LSTM, the first neural network architecture that introduces the concept of memory cell [6], on agents actions to extract more information from frames.

1.1 Deep Q-Learning

The first algorithm proposed by DeepMind was Deep Q-Learning, based on Q-Learning with experience replay [3], with this technique they save the last N experience tuples in replay memory This approach is in some respects limited since the memory buffer does not differentiate important transitions and always overwrites with recent transitions due to the finite memory size N.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Fig. 1: Deep Mind DQN algorithm with experience replay [3].

1.2 Deep Double Q-Learning

Hado et al [7] propose the idea of Double Q-learning is to reduce over estimations by decomposing the max operation in the target into action selection and action evaluation.

– **DQN Model**:

$$Y_t = R_{t+1} + \gamma max Q(S_{t+1}; a_t; \theta_t)$$

– **DDQN Model**:

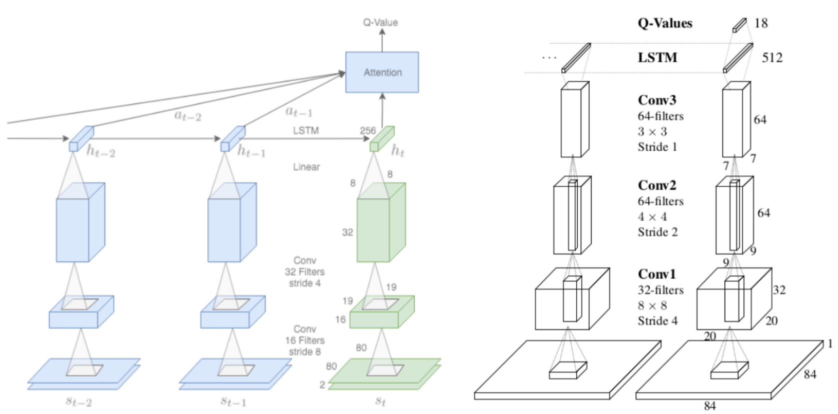
$$Y_t = R_{t+1} + \gamma Q(S_{t+1}; argmaxQ(S_{t+1}; a_t; \theta_t); \theta_t^1)$$

Where:

– • a_t represents the agent.
– • θ_t are the parameters of the network.
– • Q is the vector of action values.
– • Y_t is the target updated resembles stochastic gradient descent.
– • γ is the discount factor that trades off the importance of immediate and later rewards.
– • S_t is the vector of states.
– • R_{t+1} is the reward obtained after each action.

1.3 Deep Recurrent Q-Learning

Mathew et al [4] have been shown to be capable of learning human-level control policies on a variety of different Atari 2600 games. So they propose a DRQN algorithm which convolves three times over a single-channel image of the game screen. The resulting activation functions are processed through time by an LSTM layer.



(a) DQRNN (b) DQRN

Fig. 2: Deep Q-Learning with Recurrent Neural Networks model (a) [5] and Deep Recurrent Q-Learning model (b)[4].

1.4 Deep Q-Learning with Recurrent Neural Networks

Chen et al. [5] says DQN are limited, so they try toimprove the behavior of the network using Recurrent networks (DRQN) using LSTM in the networks to take better advantage of the experience generated in each action.

2 Proposed model

We implement the CNN proposed by Chen et al. [5] with some variations in the last layers and using ADAM error. The first attempt was a simple CNN with 3 Conv 2D layers, with the Q-Learning algorithm, we obtain a slow learning process for easy games like SpaceInvaders or Pong and very low accuracy in complicated games like Beam Rider or Enduro. Then, we try modifying using Dense 512 and 128 networks at last layer with linear activation and relu, adding a LSTM layer with activation tanh.

In Table ?? we present our Hyperparameters using in our models, we denote this list of hhyperparameters as the better set (in our case). We run models over an NVIDIA GeForce GTX 950 with Memory 1954MiB using Tensorflow, Keras and GYM (Atari library) for python. We implement DDQN, DRQN, DQN and our proposed to combine DRQN with Double Q-Learning [7] algorithm using LSTM.

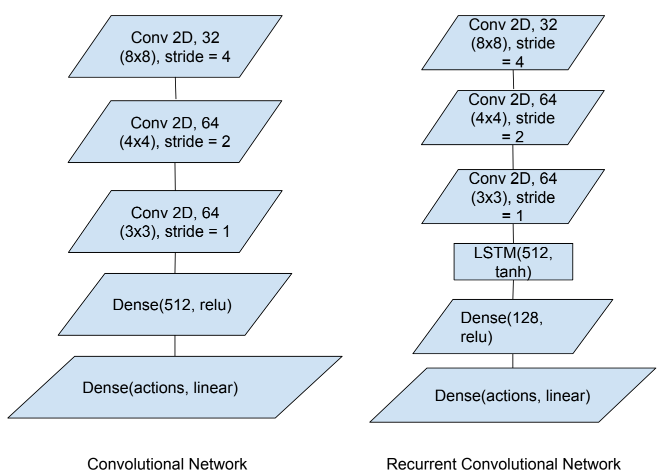


Fig. 3: Convolutional Networks used in our models.

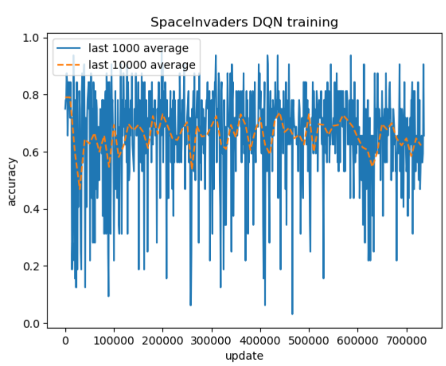
3 Results

Our experiments is build over Atari Learning Enviroment (ALE) [8] which serve us as an evaluation platform for our algorithm and allow us to compare with DQN, DDQN and DRQN. After to run our algorithms using 10M (10 millions) episodes, we obtain results for each model in each respective game. We get best scores for the 4 games mentioned above (SpaceInvaders, Enduro, Beam Rider and Pong).

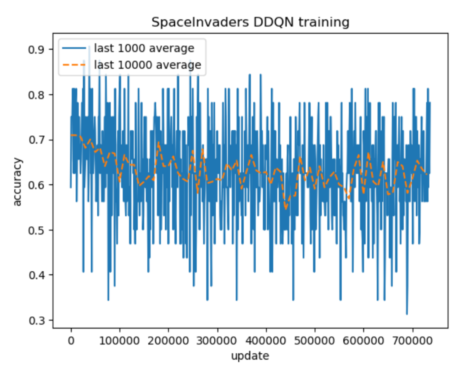
Models and respective Scores				
Model	SpaceInvaders	Enduro	Pong	Beam Rider
DQN	1450	1095	65	349
DRQN	1680	885	39	594
DDQN	2230	1283	44	167
DRDQN	2450	1698	74	876

Table 1: Results Scores of Space Invaders, Enduro, Pong and Beam Rider.

We compare with Volodymyr et al. [9] Letter about best scores form games obtained by DQN agents and professionals gamers (humans) to verify correct behavior of learning process, we measure accuracy based on Q-tables from the agent and DL algorithm (Double Q-Learning) extracting information from frames with the Convolutional Neural Networks (See Fig. 4).



(a) DQN Accuracy



(b) DDQN Accuracy

Fig. 4: DDQN vs DQN Accuracy.

4 Conclusions

We Present a model based on DRQN and Double Q-Learning combined to get a better performance in some games, using LSTM and CNN to analyze frames. We notice that each method could be good for an specific Atari game and other similar games but not for all. but can be improved using different CNN and get more information from the frames in each batch iteration.

5 Acknowledgements

This work was supported by grant 234-2015-FONDECYT (Master Program) from CienciActiva of the National Council for Science,Technology and Technological Innovation (CONCYTEC-PERU).

6 Appendix: Hyperparameters

Note: Y means Target Network.

List of Hyperparameters		
Iterations	10 000000	number of batch iterations to the learning process
miniBatch size	32	number of experiences for SGD update
Memory buffer size	900000	SGD update are sampled from this number of most recent frames
Learning Rate	0.00025	learning rate used by RMS Propagation
Training Frequency	4	Repeat each action selected by the agent this many times
Y Update Frequency	40000	number of parameter updates after which the target network updates
Update Frequency	10000	number of actions by agent between successive SGD updates
Replay start size	50000	The number of Replay Memory in experience
Exploration max	1.0	Max value in exploration
Exploration min	0.1	Min value in exploration
Exploration Steps	850000	The number of frames over which the initial value of e reaches final value
Discount Factor	0.99	Discount factor γ used in the Q-learning update

Table 2: Hyperparameters used in models

References

1. Richard S. Sutton, "Reinforcement Learning Architectures".
2. Richard Sutton and Andrew Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.
3. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Daan Wierstra, Alex Graves, Ioannis Antonoglou, Martin Riedmiller, "Playing Atari with Deep Reinforcement Learning", In NIPS Deep Learning Workshop 2013.

4. Matthew Hausknecht and Peter Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs", In AAAI Fall Symposium Series 2015.
5. Clare Chen, Vincent Ying, Dillon Laird, "Deep Q-Learning with Recurrent Neural Networks".
6. Hochreiter and Schmidhuber. Long short-term memory. Neural Comput. 9(8):1735-1780.
7. Hado van Hasselt, Arthur Guez and David Silver, "Deep Reinforcement Learning with Double Q-learning".
8. M. Bellemare, Y. Naddaf, J. Veness and M. Bowling, The arcade learning enviroment. An evaluation platform for general agents. In Journal of Artificial Intelligence Research, 47:253-279, 2013
9. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis, "Human-level control through deep reinforcement learning".